HIDDEN PHYSICS MODELS:
MACHINE LEARNING OF NONLINEAR PDEs

Samuele Brunati　　Marta Cerri　　Martina Ciancarelli

Exam project for the course of *Computational Statistics* (prof. Andrea Manzoni)
A.Y. 2022/23

SUMMARY AND OBJECTIVES

1. MODEL SETUP
2. REPRODUCE THE RESULTS
3. QUESTIONS
4. CODE IMPLEMENTATION
5. CONLUSIONS

ESTIMATE the PARAMETERS of a given (physics is known) equation

Arising questions and exploration of the model

Code to generalize the analysis to other EXAMPLES

Comparison with alternative approaches

[1] M.RAISSI, G.KARNIADAKIS, Hidden Physics Models: Machine Learning of Nonlinear Partial Differential Equations, arXiv:1708.00588, 2017

PROBLEM SETUP

GOAL : given a non linear PDE $h_t + \mathcal{N}_x^\lambda h = 0, \; x \in \Omega, \; t \in [0,T]$

and 2 observed snapshots $\{\boldsymbol{x}^{n-1}, \boldsymbol{h}^{n-1}\}$, $\{\boldsymbol{x}^{n}, \boldsymbol{h}^{n}\}$

what are the parameters λ that best describe the observed data?

IDEA : assume Δt small enough ➡ apply backward Euler $h^n + \Delta t \mathcal{N}_x^\lambda h^n = h^{n-1}$ ➡ $\mathcal{L}_x^\lambda h^n = h^{n-1}$

MODEL : $h^n(x) \sim \mathcal{GP}(0, k(x, x', \theta))$

$$k^{n,n} = k, \qquad k^{n,n-1} = \mathcal{L}_{x'}^\lambda k,$$
$$k^{n-1,n} = \mathcal{L}_x^\lambda k \qquad k^{n-1,n-1} = \mathcal{L}_x^\lambda \mathcal{L}_{x'}^\lambda k$$

multi-output Gaussian process : $\begin{bmatrix} h^n \\ h^{n-1} \end{bmatrix} \sim \mathcal{GP}\left(0, \begin{bmatrix} k^{n,n} & k^{n,n-1} \\ k^{n-1,n} & k^{n-1,n-1} \end{bmatrix}\right)$

LEARNING PROCESS

Likelihood: $p(\boldsymbol{h}|\theta,\lambda,\sigma^2) = \mathcal{N}(\boldsymbol{0},\boldsymbol{K})$

$$\boldsymbol{h}^n = h^n(\boldsymbol{x}^n) + \boldsymbol{\epsilon}^n \text{ and } \boldsymbol{h}^{n-1} = h^{n-1}(\boldsymbol{x}^{n-1}) + \boldsymbol{\epsilon}^{n-1} \text{ with } \boldsymbol{\epsilon}^n \sim \mathcal{N}(0,\sigma^2 I)$$

MINIMIZATION PROBLEM : minimize the negative log marginal likelihood

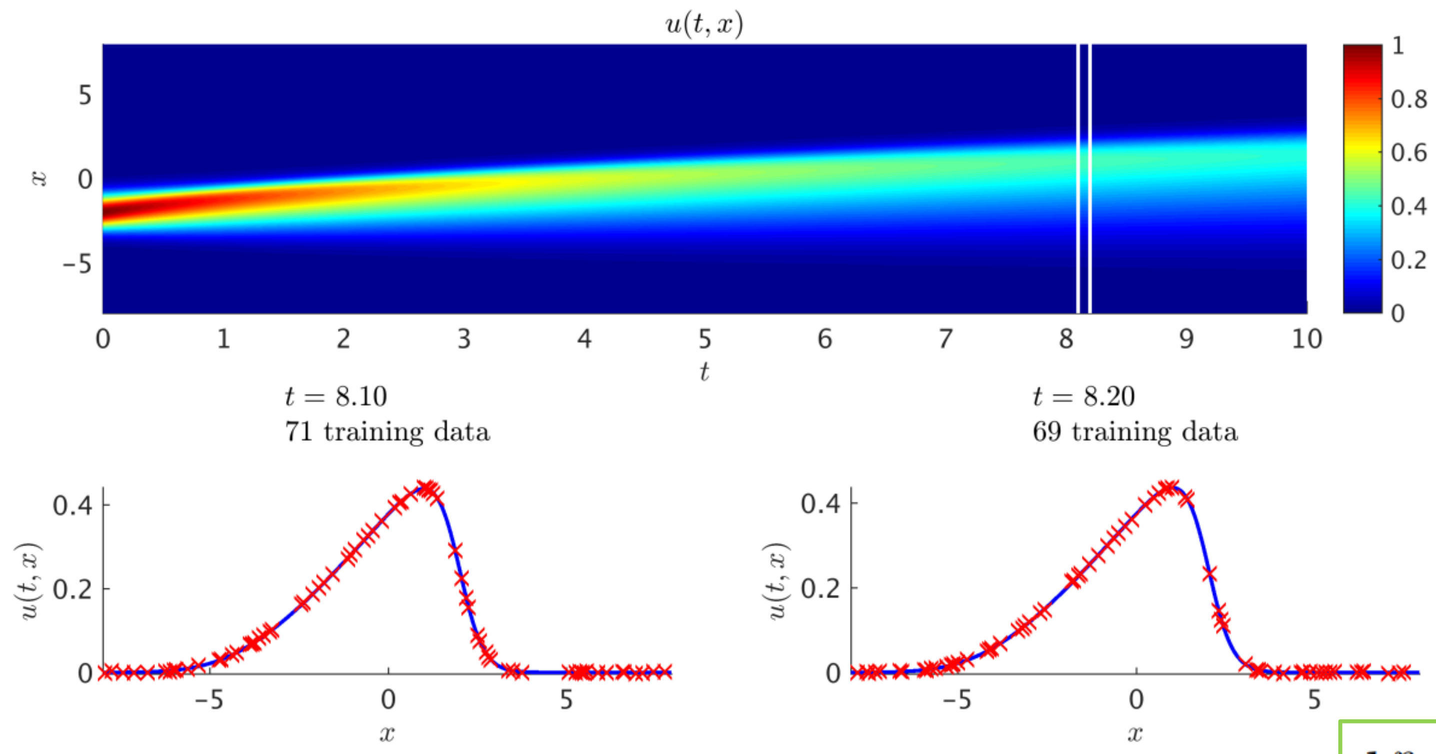
$$-\log p(\boldsymbol{h}|\theta,\lambda,\sigma^2) = \frac{1}{2}\boldsymbol{h}^T\boldsymbol{K}^{-1}\boldsymbol{h} + \frac{1}{2}\log|\boldsymbol{K}| + \frac{N}{2}\log(2\pi)$$

Fit the training data

Inverse matrix
Non convex function

Penalizes model compexity

EXAMPLES



Correct PDE	$u_t + uu_x - 0.1u_{xx} = 0$
Identified PDE (clean data)	$u_t + 1.028uu_x - 0.101u_{xx} = 0$
Identified PDE (1% noise)	$u_t + 1.017uu_x - 0.094u_{xx} = 0$

Equations

- **Burgers'**
- KdV Equation
- Kuramoto-Sivashinsky
- Nonlinear Schrödinger
- Navier-Stokes

$$h^n + \Delta t \mathcal{N}_x^{\lambda} h^n = h^n + \Delta t(\lambda_1 h^n h_x^n - \lambda_2 h_{xx}^n)$$

$$\mathcal{L}_x^{\lambda} h^n = h^n + \Delta t(\lambda_1 h^{n-1} h_x^n - \lambda_2 h_{xx}^n)$$

Why N0 = 71, N1 = 69? Why t = 8.1?

Test for different $\mathbf{N0, N1, t}$

Data
E.g. $u_h^n \in \mathbb{R}^{256\times 101}$, solution of $u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$

Testcase
Burgers' equation

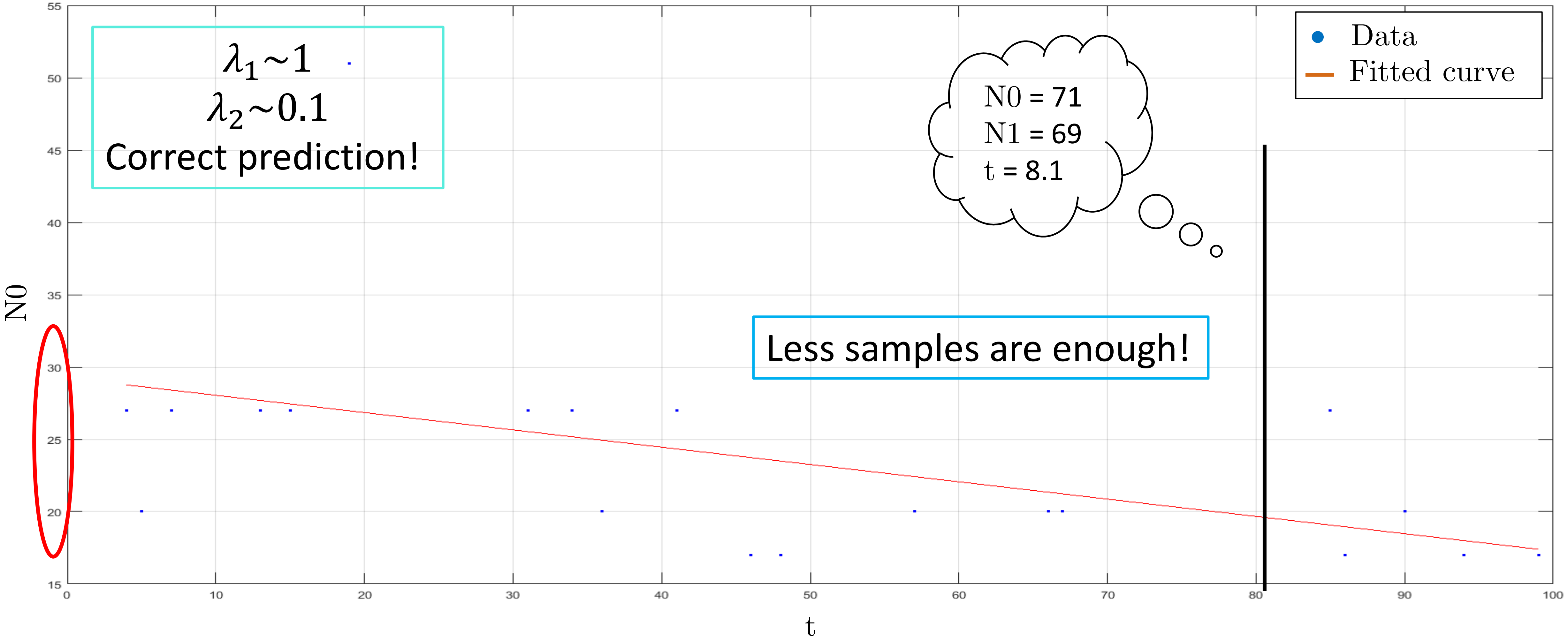
Loop(Randsample $\mathbf{N0}$, $\mathbf{N1}$ in $1{:}\,256$)

Loop(Randsample $\mathbf{t}$ in $1{:}\,101$)

Estimate λ_1 and λ_2

Compare the predictions

Compare relative increments (stagnation criterion)



$\lambda_1 \sim 1$

$\lambda_2 \sim 0.1$

Correct prediction!

Less samples are enough!

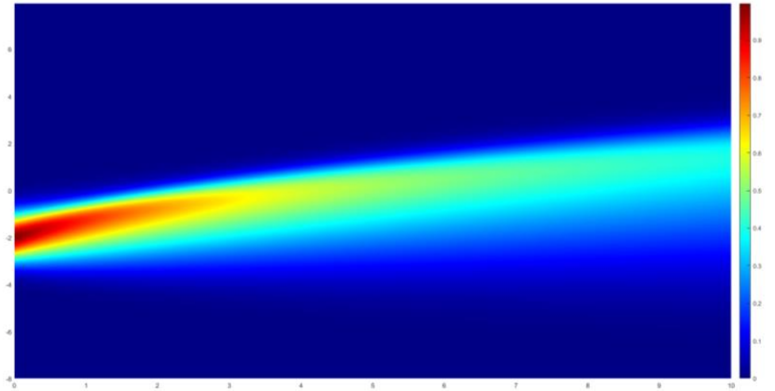
What about different λ_1 and λ_2?

What was the initial condition of the data proposed?

Testcase	
Burgers' equation	$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} = 0$

Gaussian

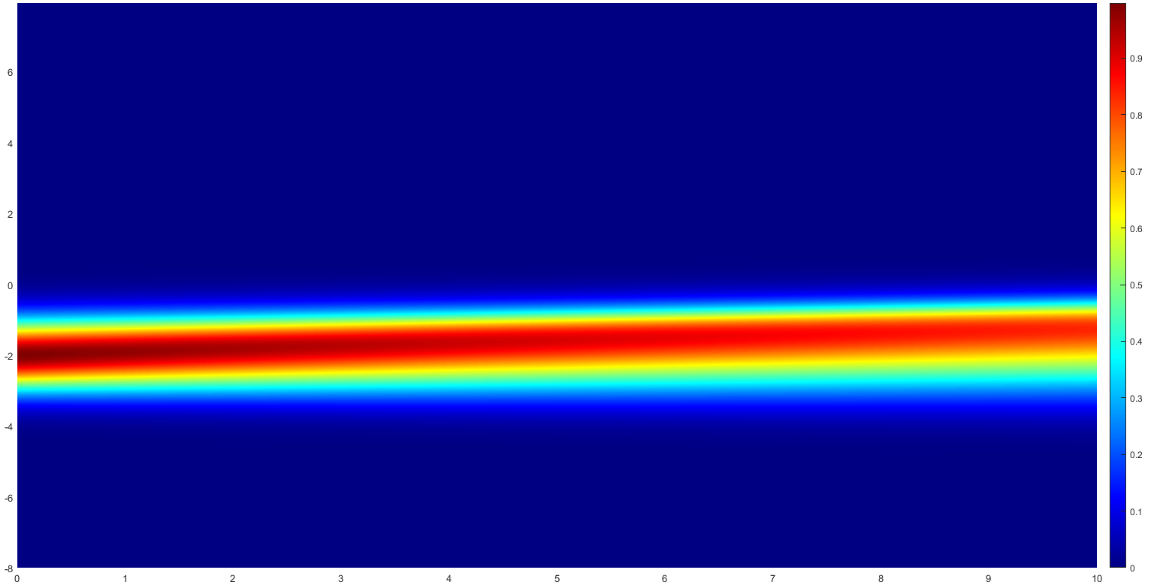
λ_1=1, λ_2=0.1



Generate new data

E.g. λ_1=0.1, λ_2=0.01

Less transport, more diffusion



Correct estimates!

What about another initial condition?

Initial condition proposed: Gaussian

Generate new data

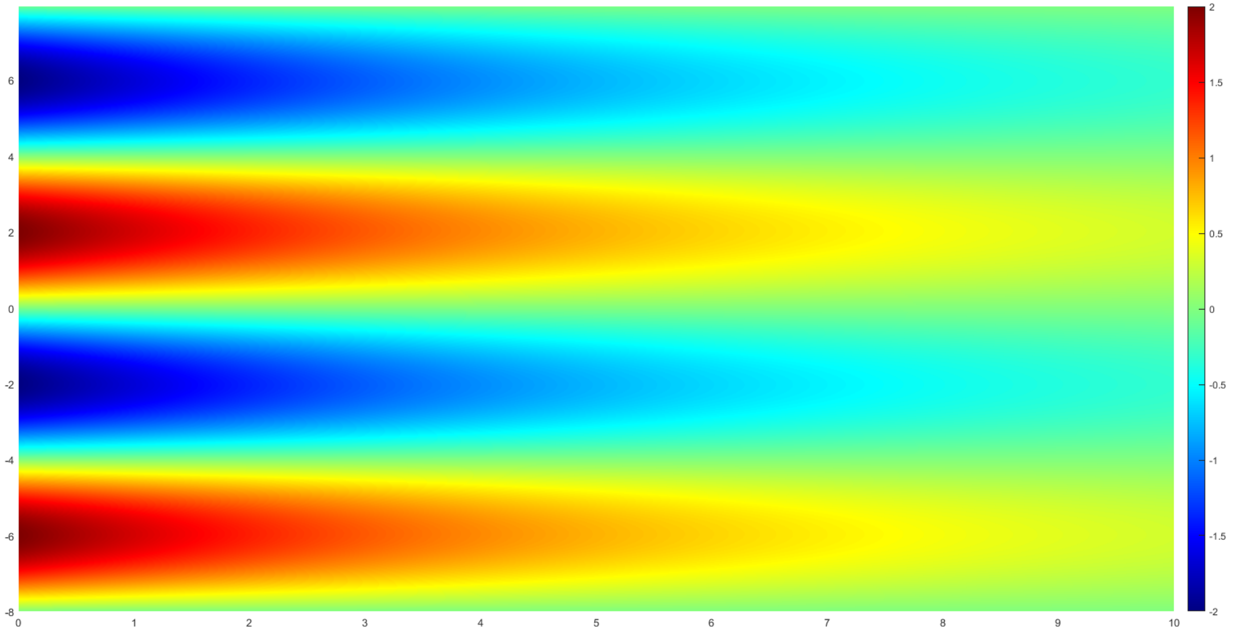
E.g. Sin, Cos, Secant, Hyperbolic Tangent, Exponential, compositions

Good estimates!

	λ_1	λ_2
First quartile	0.019464	0.30559
Median	0.01971	0.30562
Third quartile	0.020521	0.30559

Initial condition: Sinusoidal

$$u_t + 0.01uu_x - 0.3u_{xx}$$

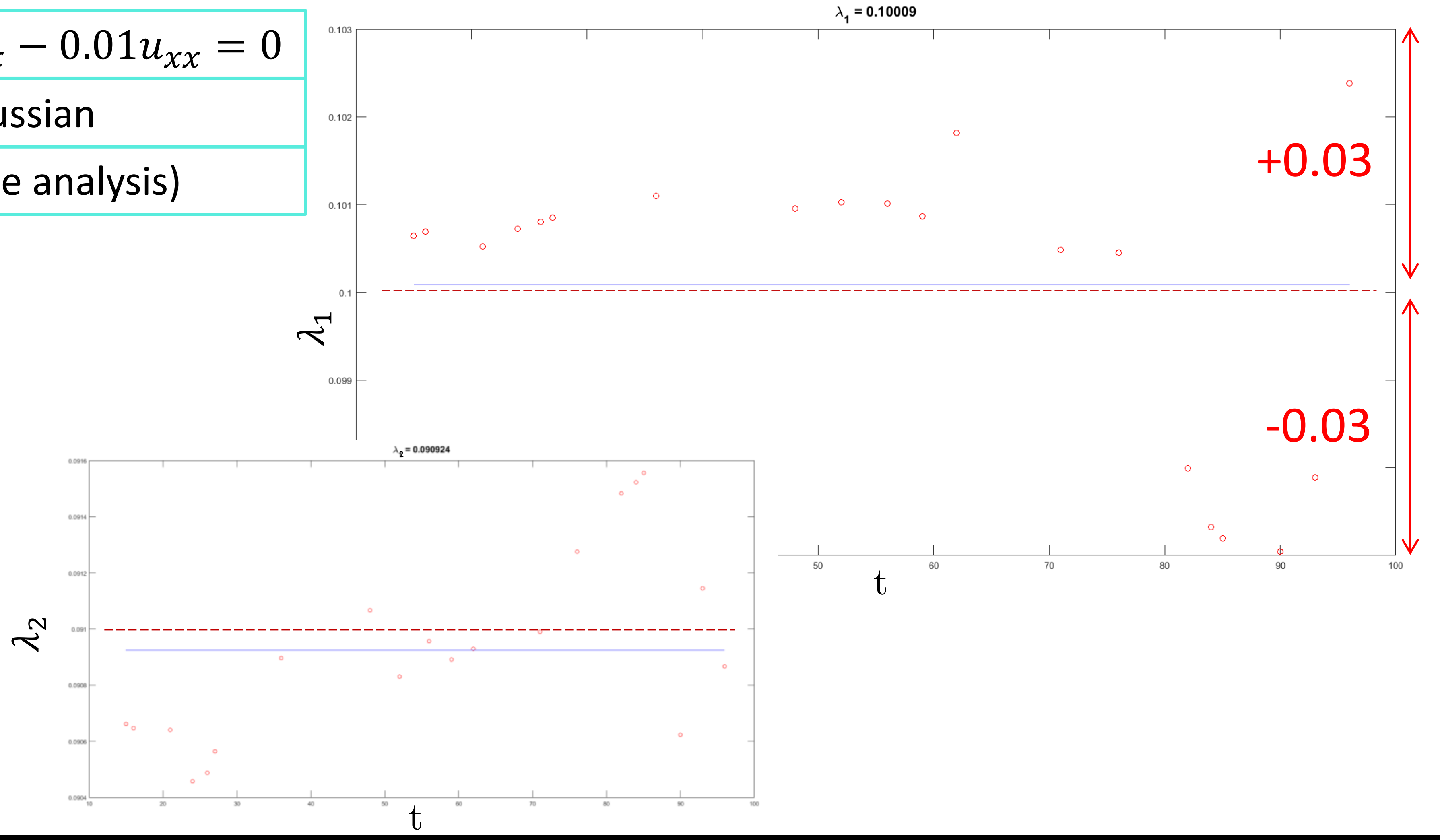


What is the impact of the noise?

Burgers' equation $u_t + 0.1uu_x - 0.01u_{xx} = 0$	
Initial condition: Gaussian	
N0 = 45, N1 = 45 (after the analysis)	

Add noise to data

Good predictions



What if we see less times and less points in space?

Burgers' equation $u_t + 0.1uu_x - 0.01u_{xx} = 0$

Initial condition: Gaussian

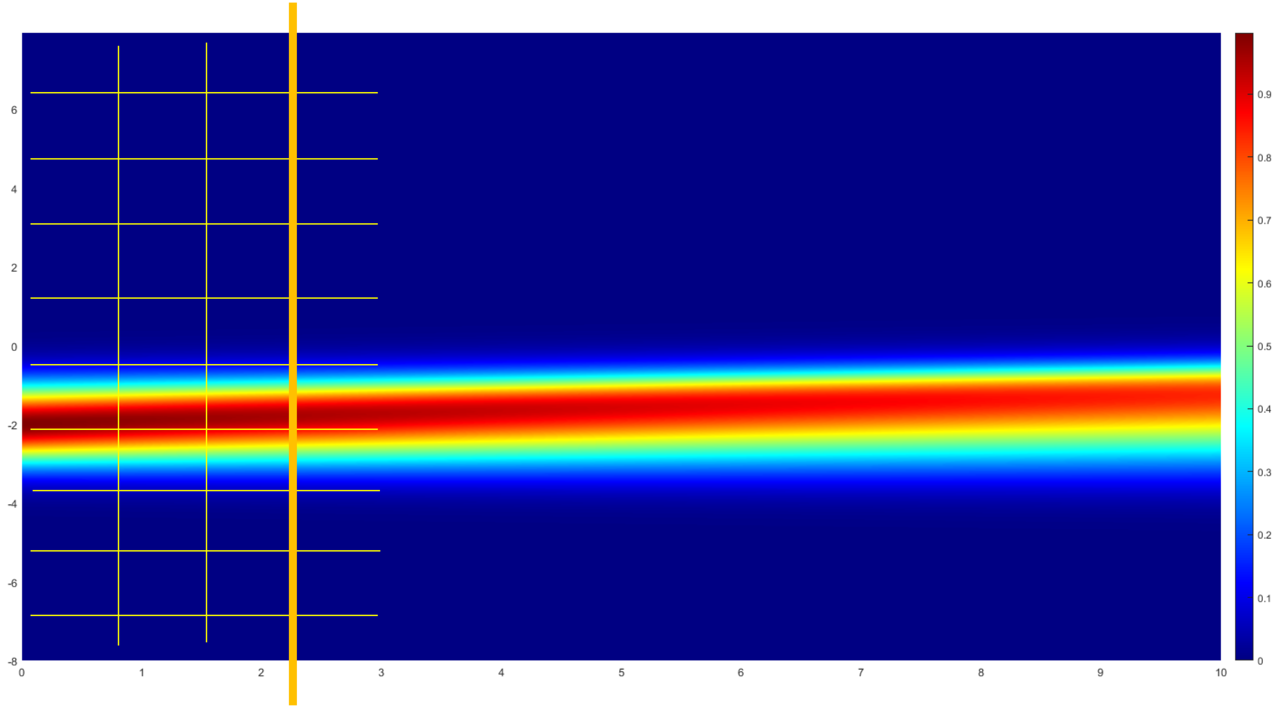
$T = 10, \Delta t = 0.1, N_h = 256$

$\boldsymbol{T} = \mathbf{3}, \Delta t = 0.1, N_h = 256$

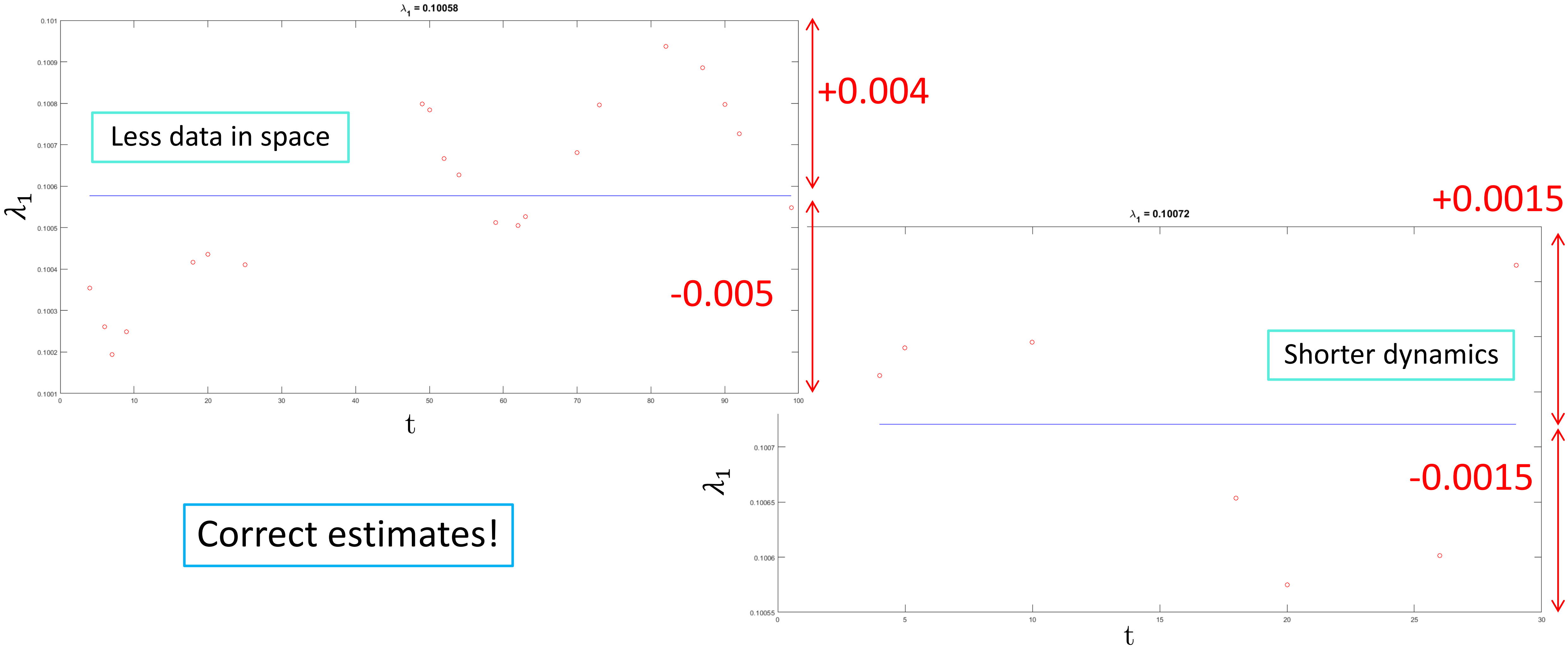
Shorter dynamics

$T = 10, \Delta t = 0.1, \boldsymbol{N_h} = \mathbf{100}$

Less data in space



What if we see less times and less points in space?



Correct estimates!

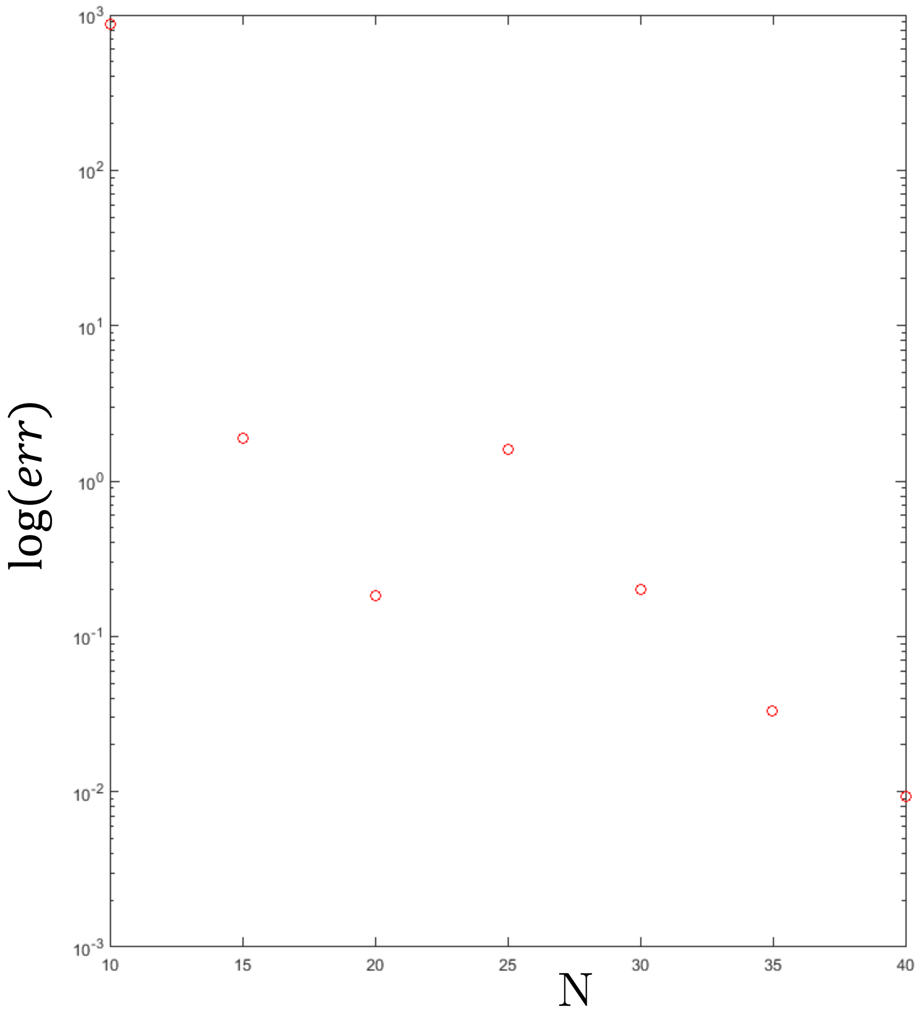
What if we use less training data?

Burgers' equation	$u_t + 0.1uu_x - 0.01u_{xx} = 0$
Initial condition: Gaussian	
N0 = 45, N1 = 45 (after the analysis)	

What if we use N < min{N0,N1} ?

$$err = \frac{|\lambda_1^N - \lambda_1^{exact}|}{|\lambda_1^{exact}|}$$

Error grows a lot!

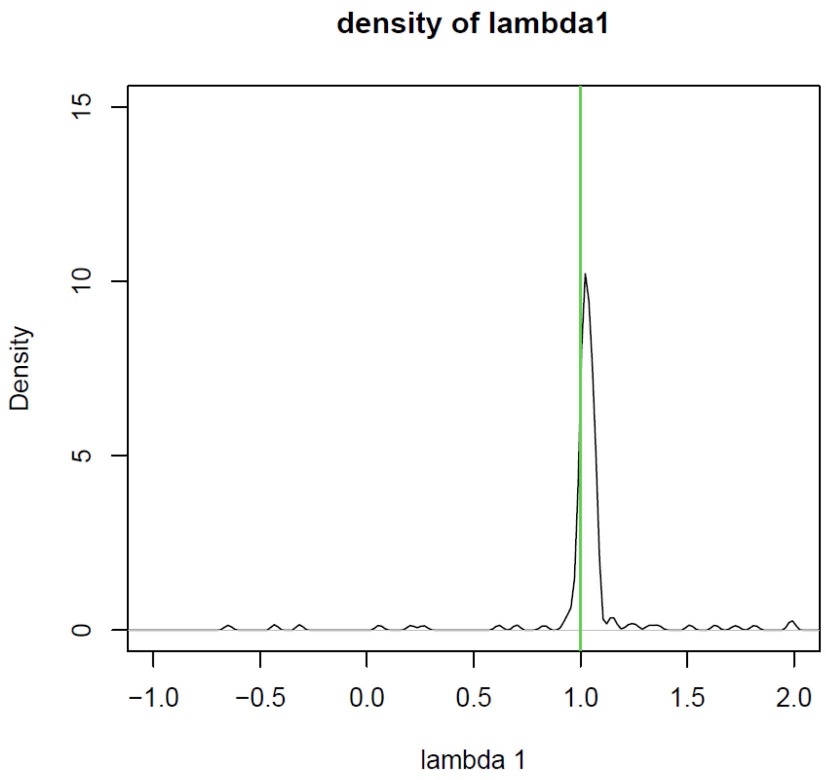


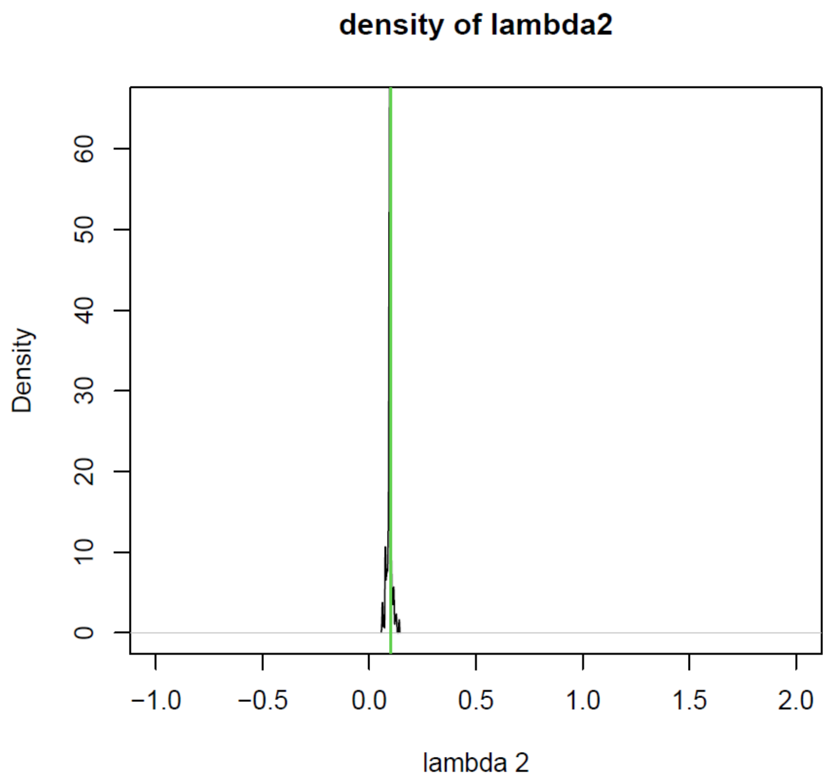
Can we recover the distribution of the parameters?

Given data: λ_1, λ_2 values

Burgers' equation

$u_t + 0.1uu_x - 0.01u_{xx} = 0$





Further questions

What about the other equations?

Code implementation

Equations

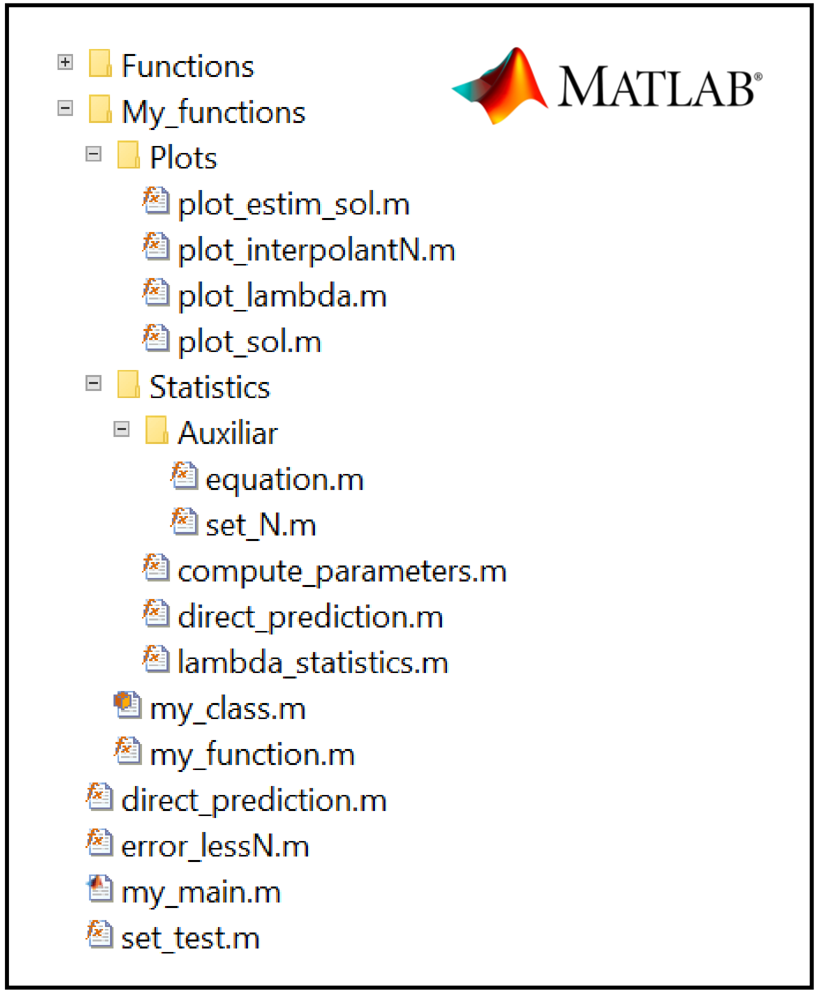
- Burgers’
- KdV Equation
- Kuramoto-Sivashinsky
- Nonlinear Schrödinger
- Navier-Stokes

General code structure

Data.mat

Testcase

1) Burgers'
2) KdV Equation
3) Kuramoto-Sivashinsky
4) Nonlinear Schrödinger
5) Navier-Stokes

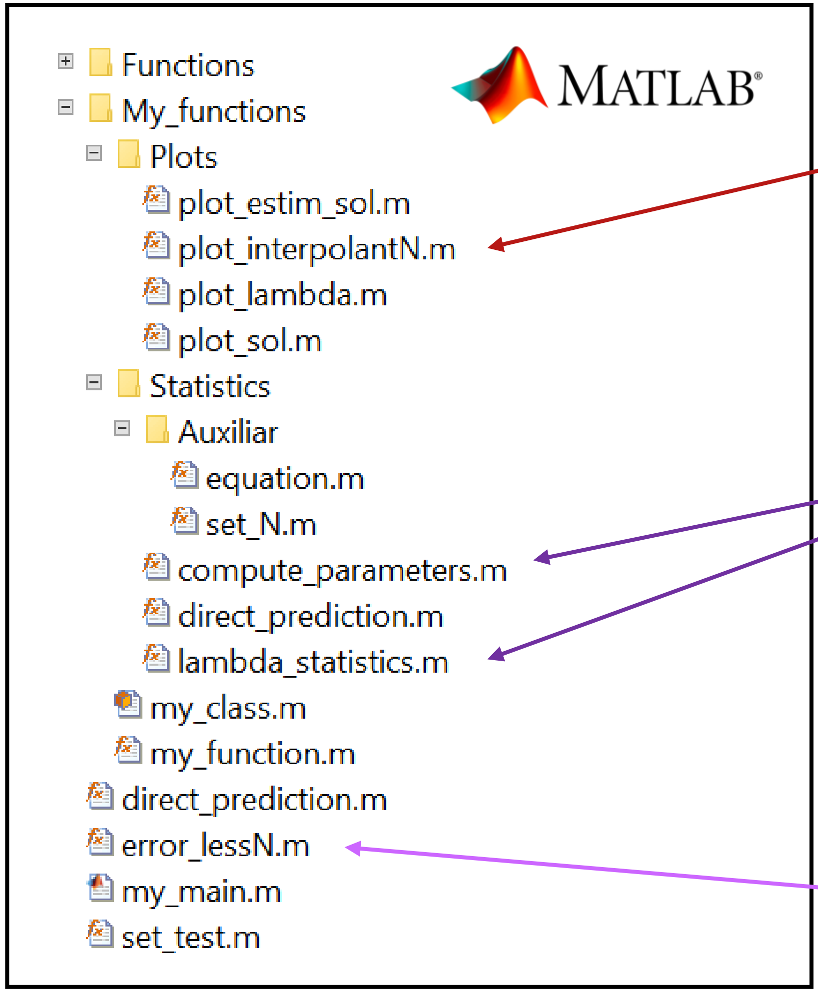


Class object

- testcase
- params
- usol
- t
- x
- λ_1
- λ_2
- λ_3

+ plot_lambda()
+ suggest_N()
+ lambda()
+ plot_solution()
+ plot_estimated_solution()

Code implementation

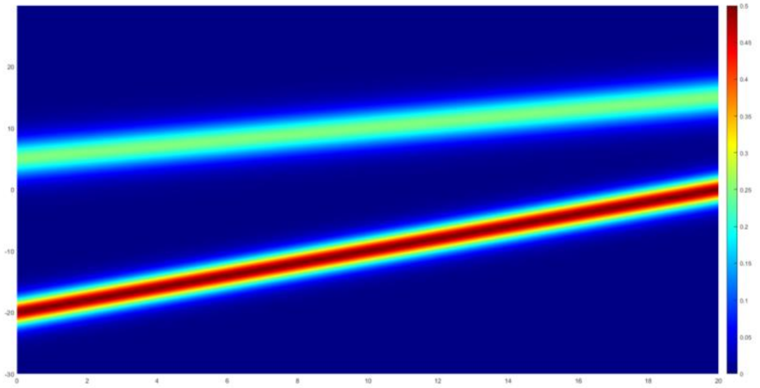


Q1: Why N0 = 71, N1 = 69? Why t = 8.1?

Q2: What about different λ_1 and λ_2?
Q3: What about another initial condition?
Q4: What is the impact of the noise?
Q5: What if we see less times and less points in space?

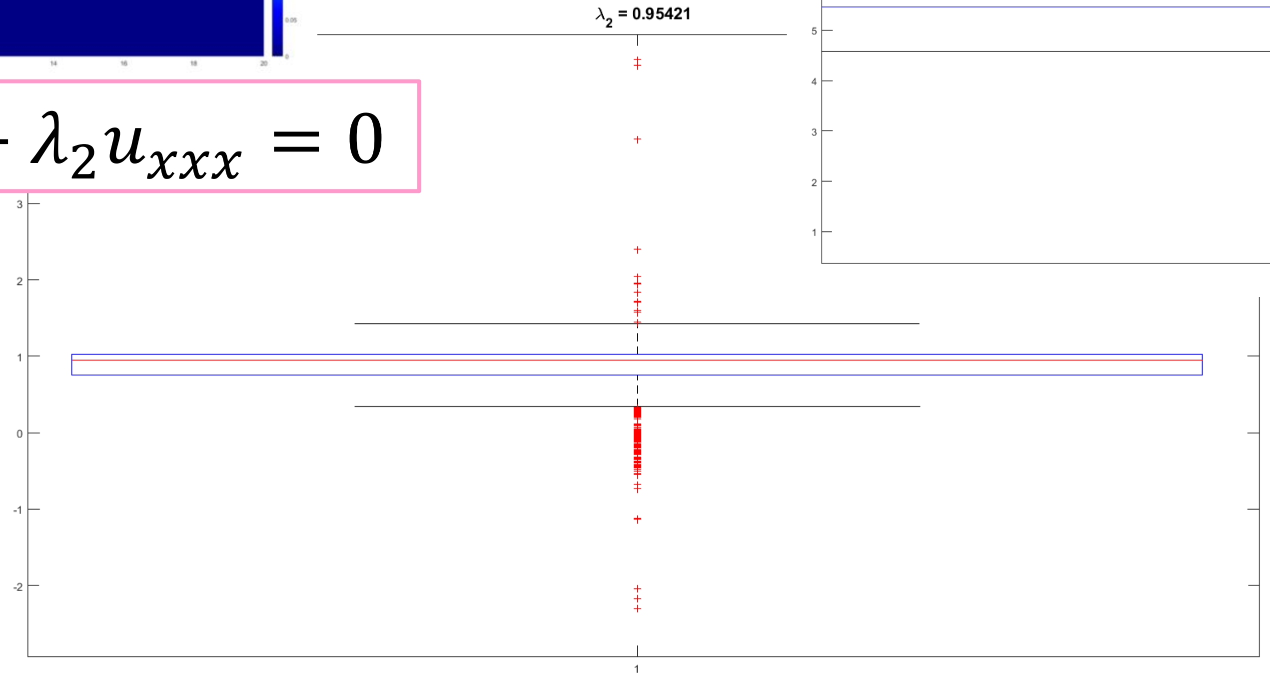
Q6: What if we use less training data?

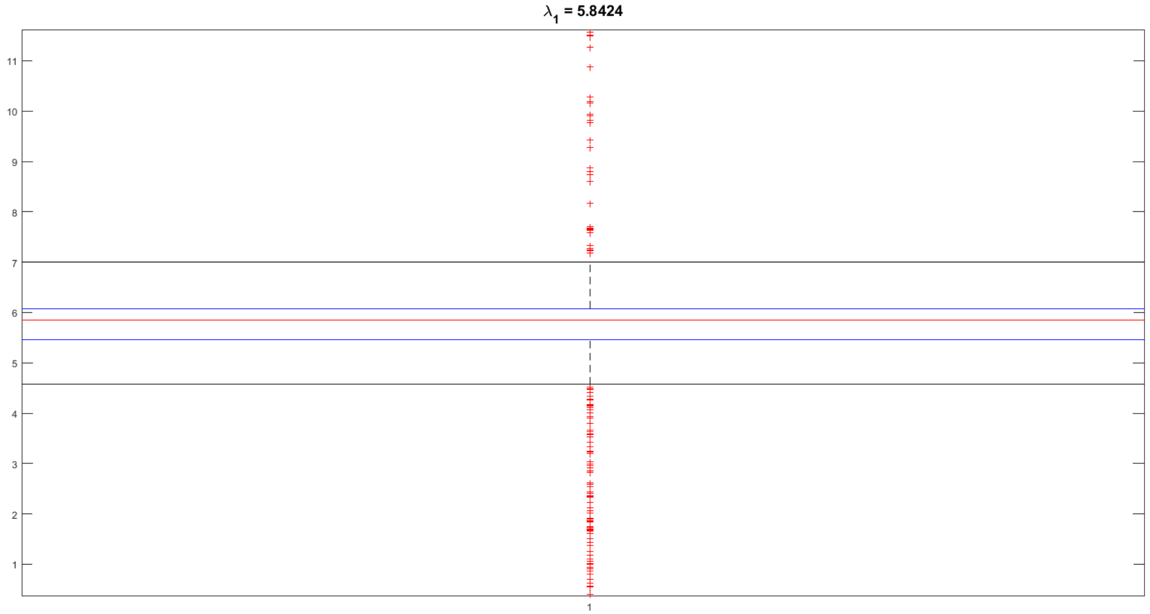
KdV Equation



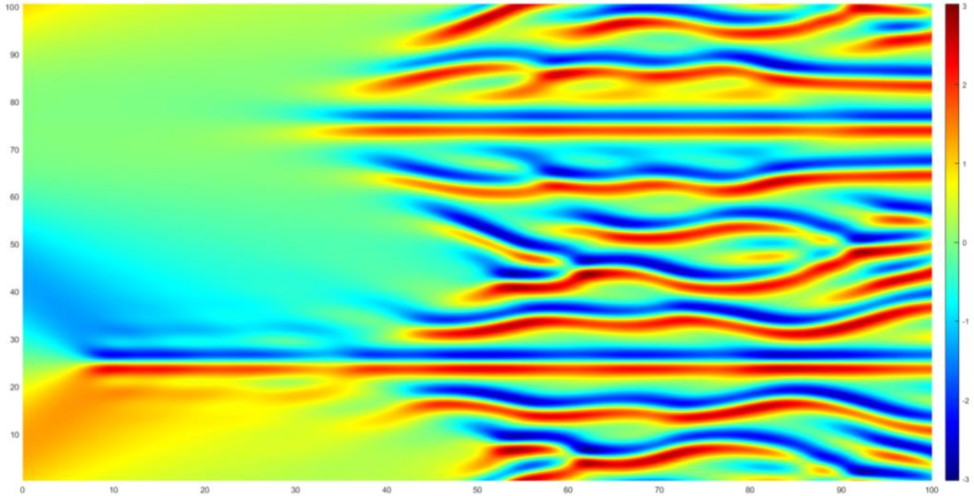
$$u_t + \lambda_1 u u_x + \lambda_2 u_{xxx} = 0$$

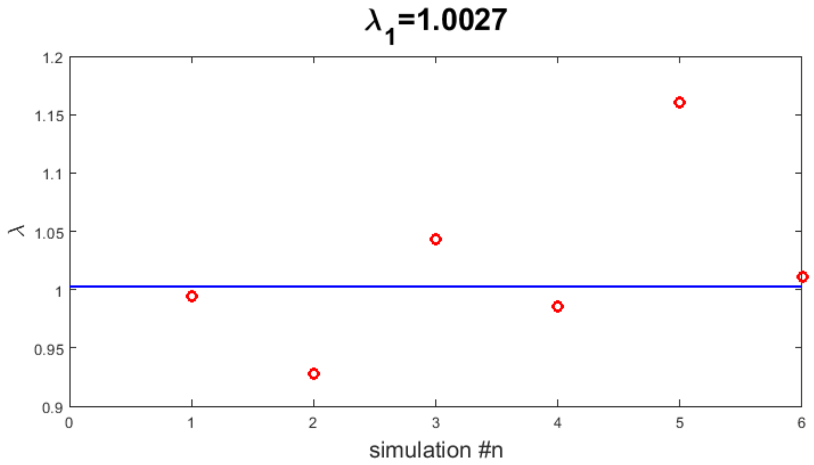
$$\lambda_1 = 6$$
$$\lambda_2 = 1$$





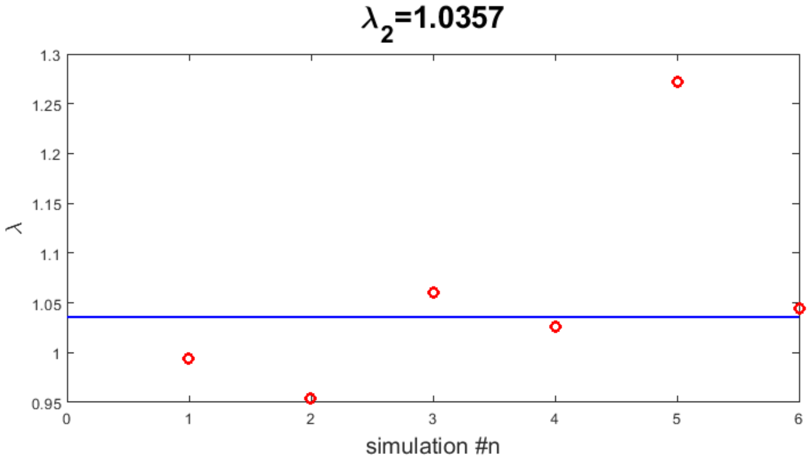
Kuramoto-Sivashinsky Equation

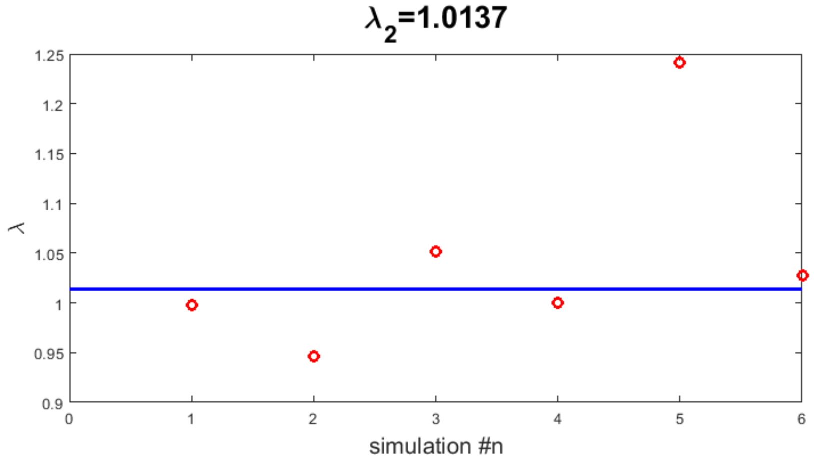




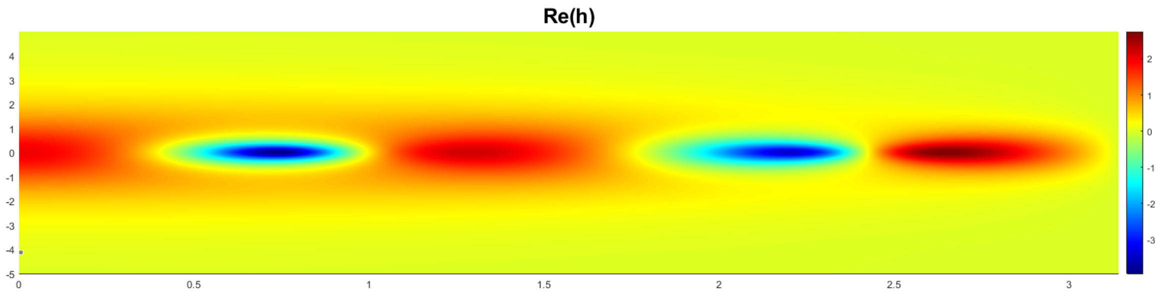
$$u_t + \lambda_1 u u_x + \lambda_2 u_{xx} + \lambda_3 u_{xxxx} = 0$$

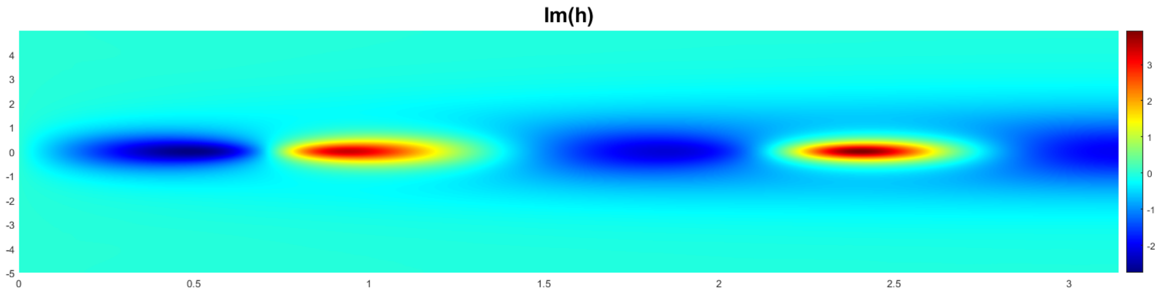
$$\lambda_1 = \lambda_2 = \lambda_3 = 1$$

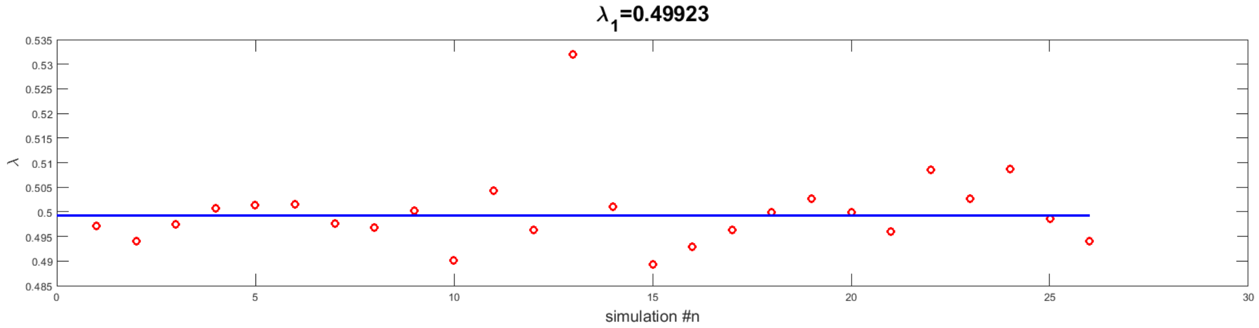




Nonlinear Schrödinger Equation

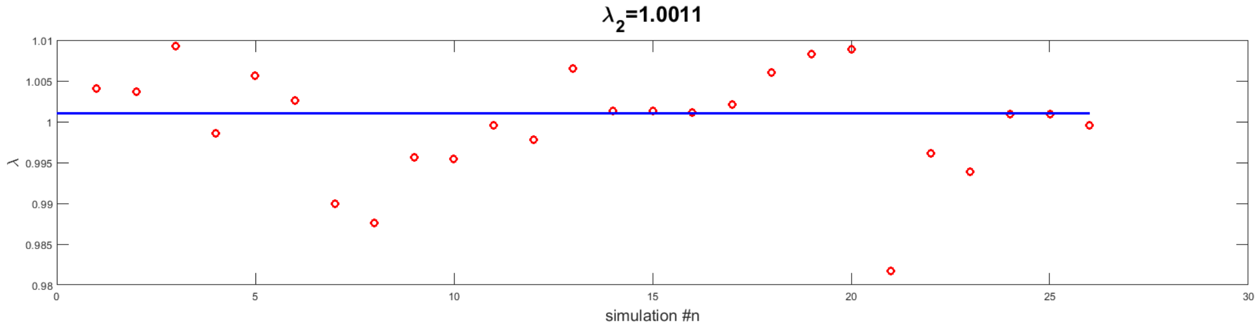




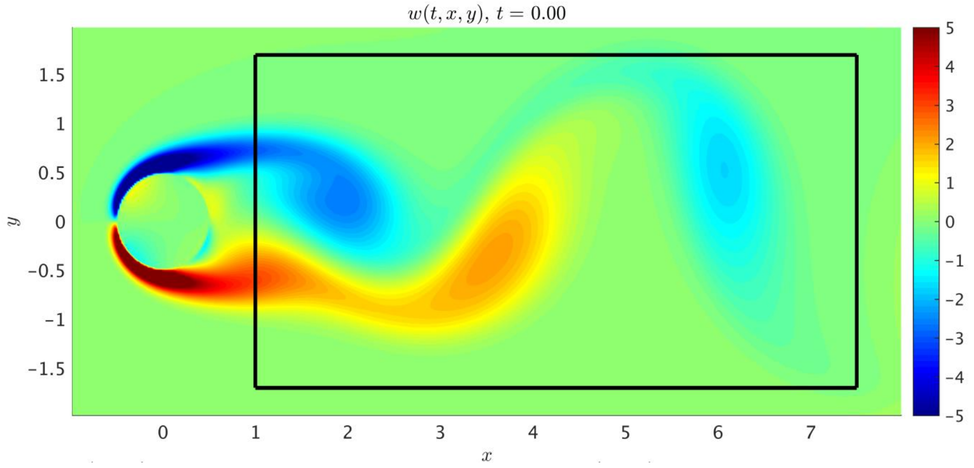


$$ih_t + \lambda_1 h_{xx} + \lambda_2 |h|^2 h = 0$$

$$\lambda_1 = 0.5$$
$$\lambda_2 = 1$$

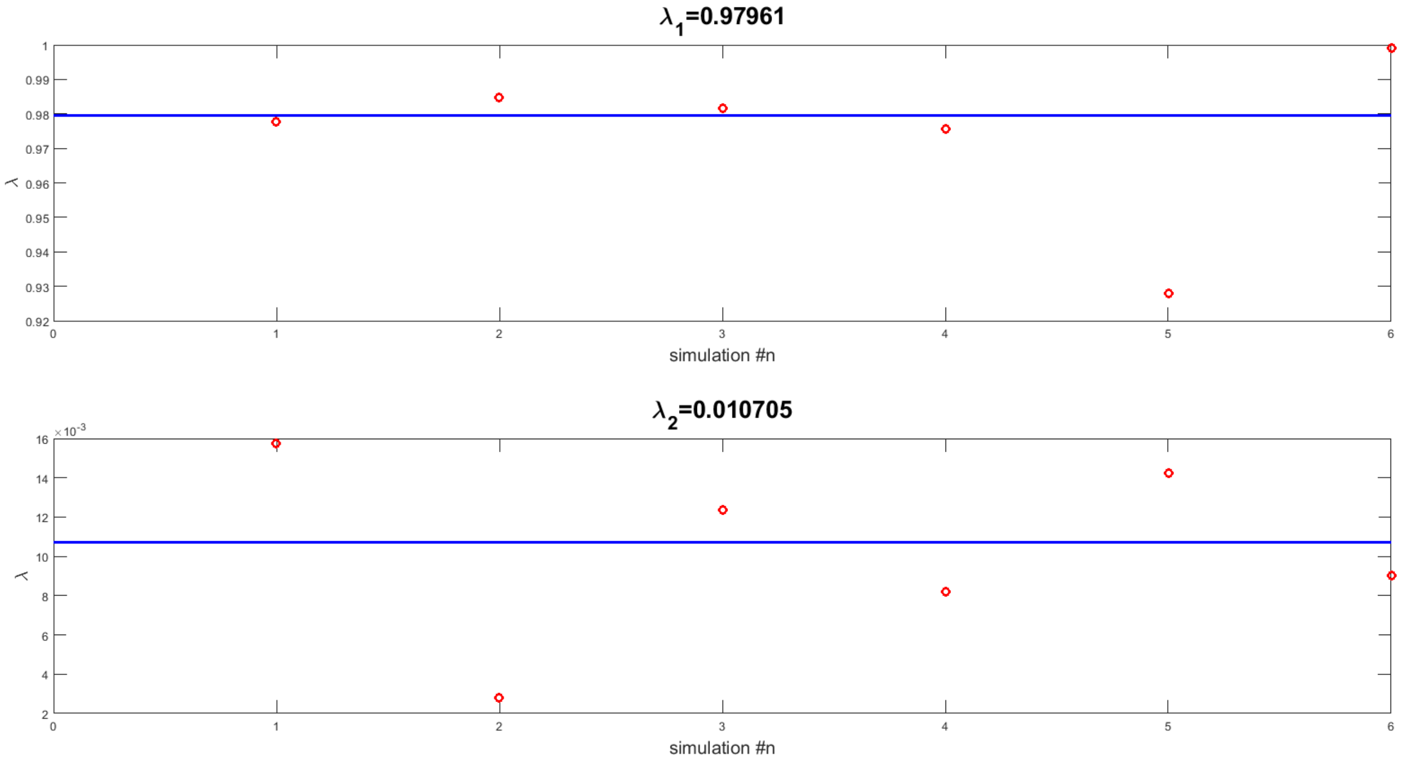


Navier-Stokes Equations



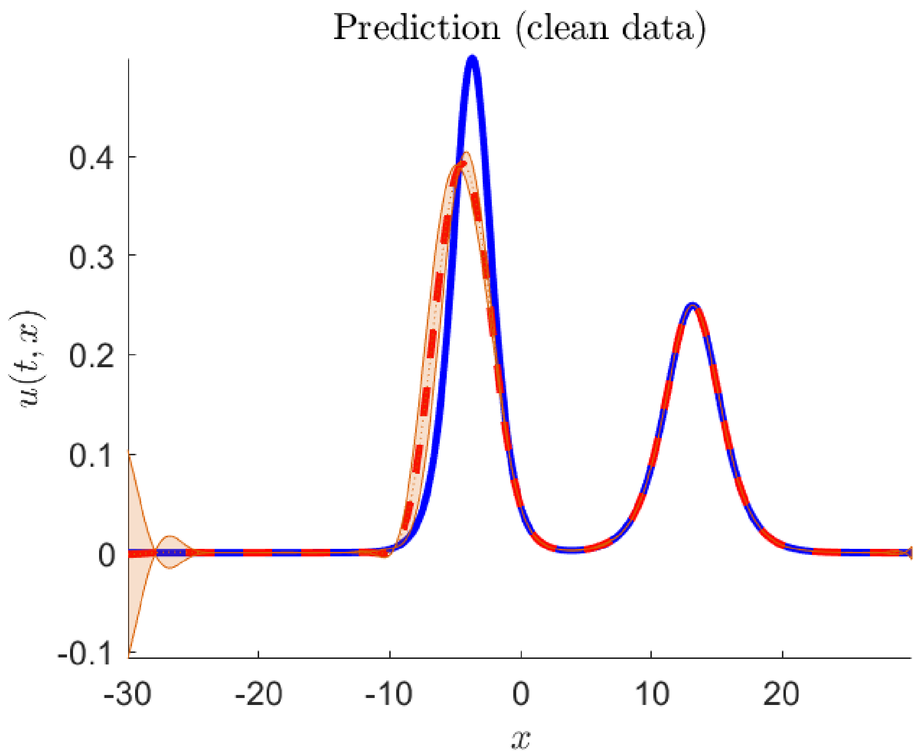
$$u_t + \lambda_1 (u \cdot \nabla) u - \lambda_2 \Delta u + \nabla p = 0$$
$$\nabla \cdot u = 0$$

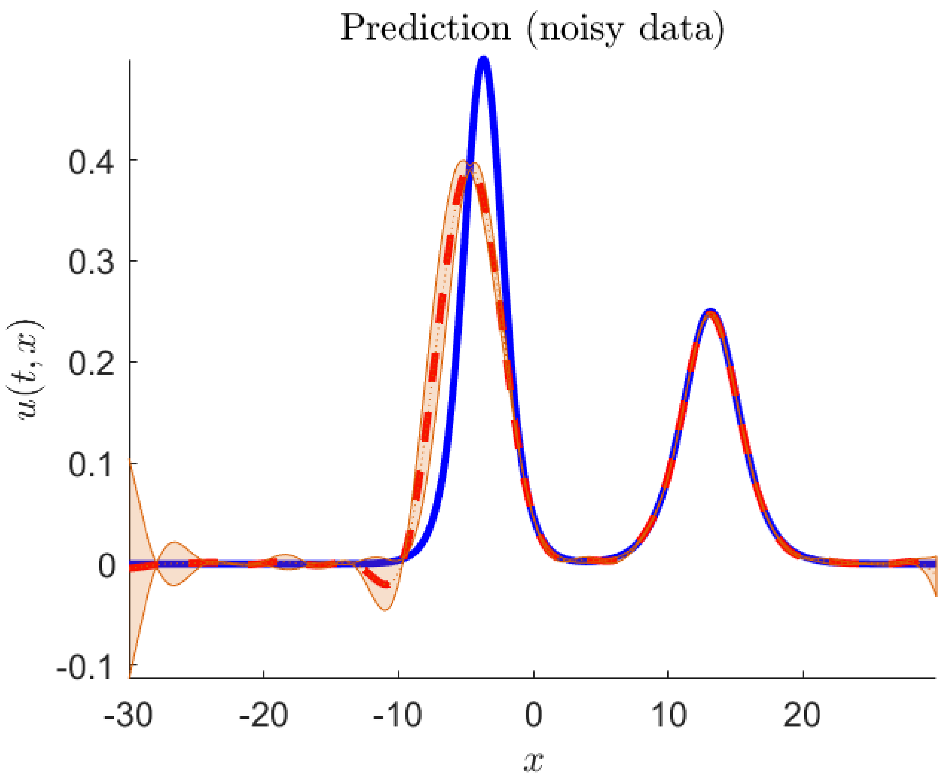
$$\lambda_1 = 1$$
$$\lambda_2 = 0.01$$



What can we say about the variance of the predictions?

Example: KdV equation
N0, N1 = 23





Is the model able to discover the physics?

Kuramoto-Sivashinsky equation

$u_h^n \in \mathbb{R}^{1024\times 251}$, solution of

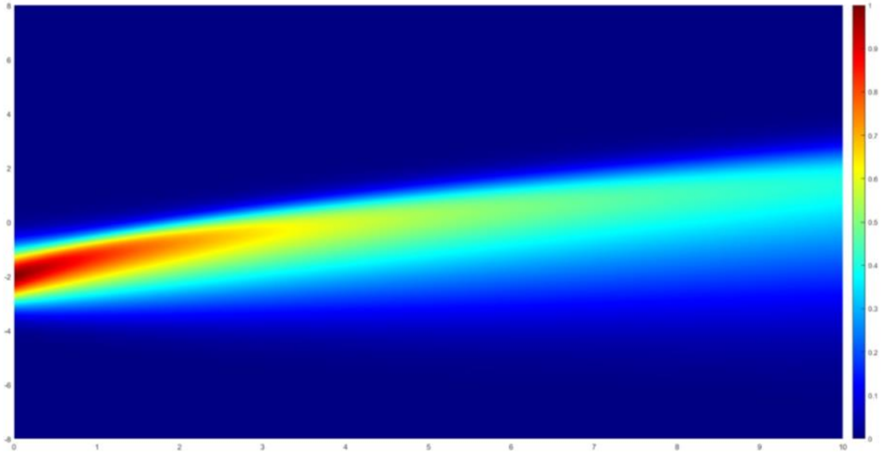
$u_t + \lambda_1 u u_x - \lambda_2 u_{xx} + \lambda_3 u_{xxxx} = 0$

Burgers' equation

Initial condition: Gaussian

λ_1=1
λ_2=0.1
λ_3=1e-16

	λ_1	λ_2	λ_3
First quartile	1,0012	0,096702	0,00013675
Median	1,0157	0,092943	0,00039946
Third quartile	1,0248	0,087183	0,00058869



Physics is not fully captured

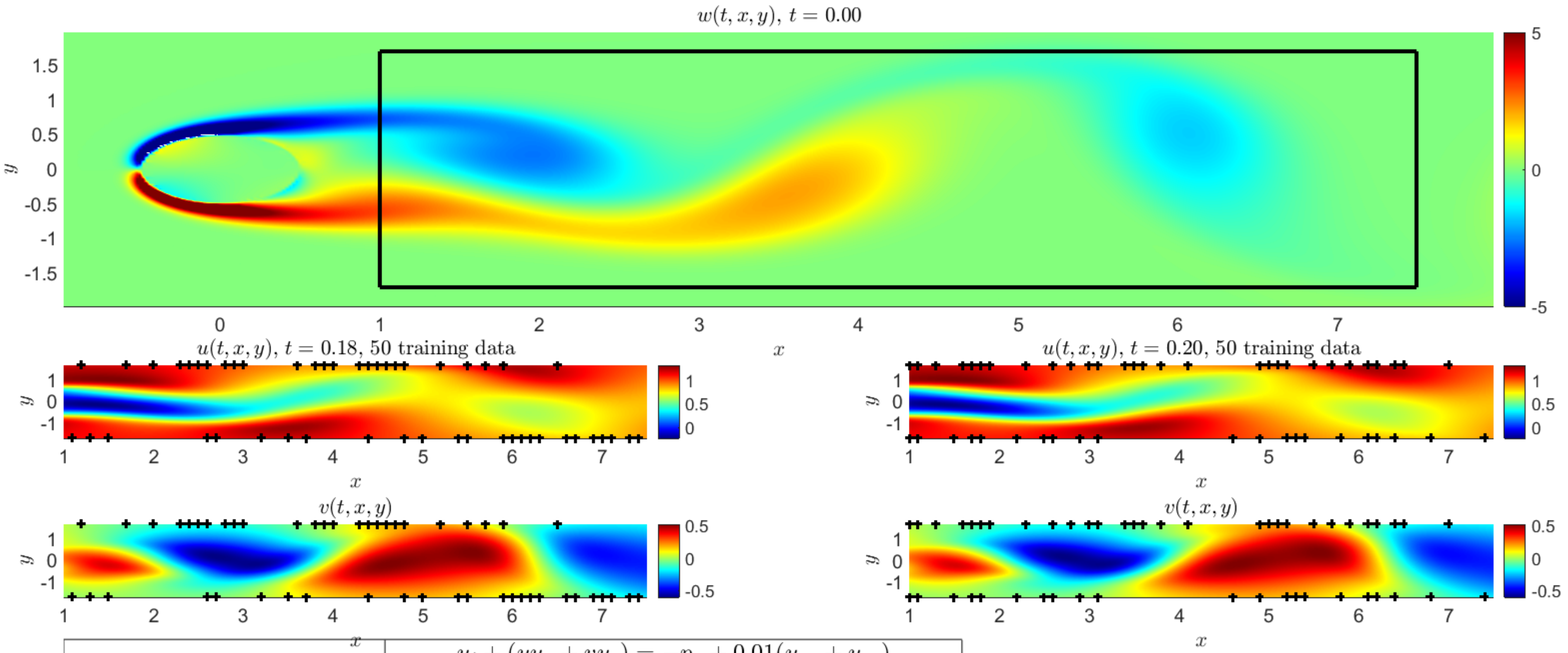
What about the location of the data?

NS Equations
$u_t + \lambda_1 (u \cdot \nabla) u - \lambda_2 \Delta u + \nabla p = 0$
$\nabla \cdot u = 0$

N = 50, t = 0.18

Training data on the boundary

Bad estimates, in particular λ_2



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.865(uu_x + vu_y) = -p_x + 0.00022(u_{xx} + u_{yy})$ $v_t + 0.865(uv_x + vv_y) = -p_y + 0.00022(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.630(uu_x + vu_y) = -p_x + 0.00103(u_{xx} + u_{yy})$ $v_t + 0.630(uv_x + vv_y) = -p_y + 0.00103(v_{xx} + v_{yy})$

What about the location of the data?

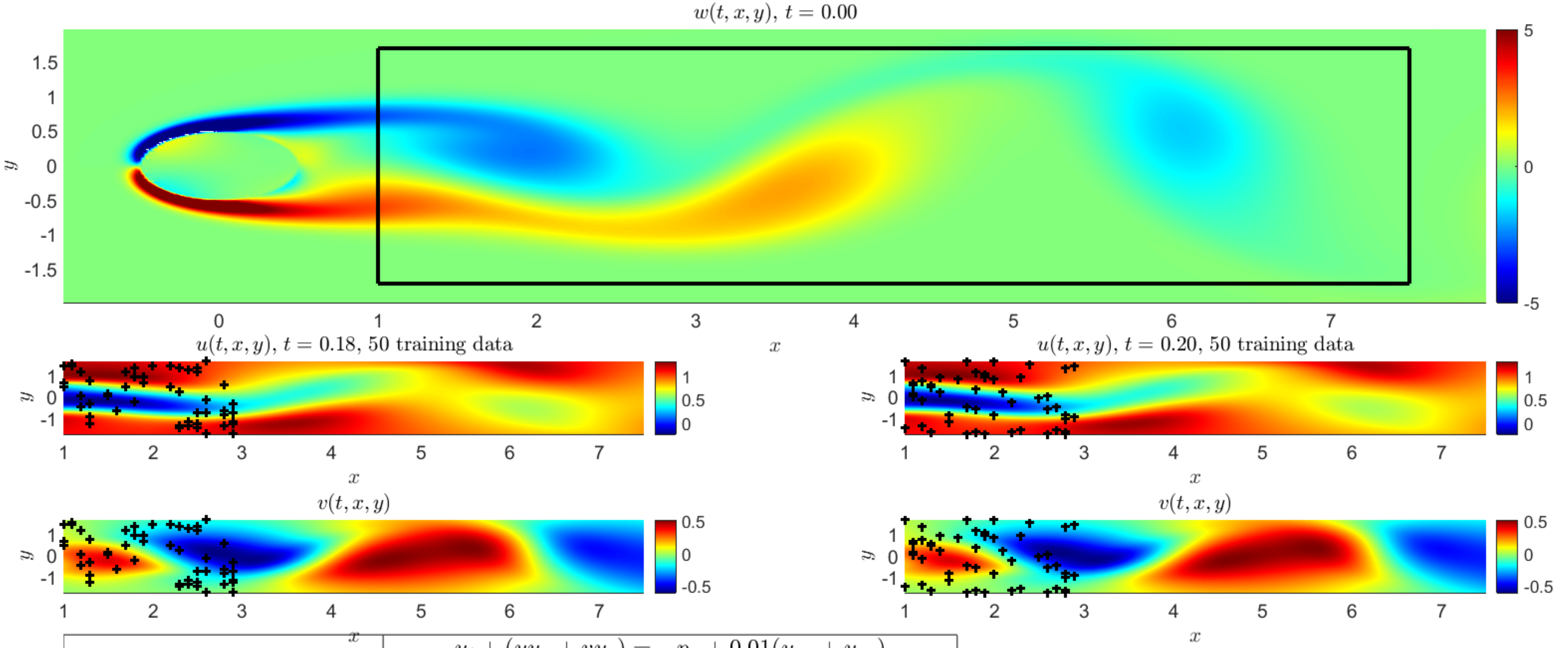
NS Equations

$u_t + \lambda_1 (u \cdot \nabla) u - \lambda_2 \Delta u$
$+ \nabla p = 0$
$\nabla \cdot u = 0$

N = 50, t = 0.18

Training data on the left side of the domain

Bad estimates, in particular λ_1



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 0.456(uu_x + vu_y) = -p_x + 0.00727(u_{xx} + u_{yy})$ $v_t + 0.456(uv_x + vv_y) = -p_y + 0.00727(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.549(uu_x + vu_y) = -p_x + 0.01071(u_{xx} + u_{yy})$ $v_t + 0.549(uv_x + vv_y) = -p_y + 0.01071(v_{xx} + v_{yy})$

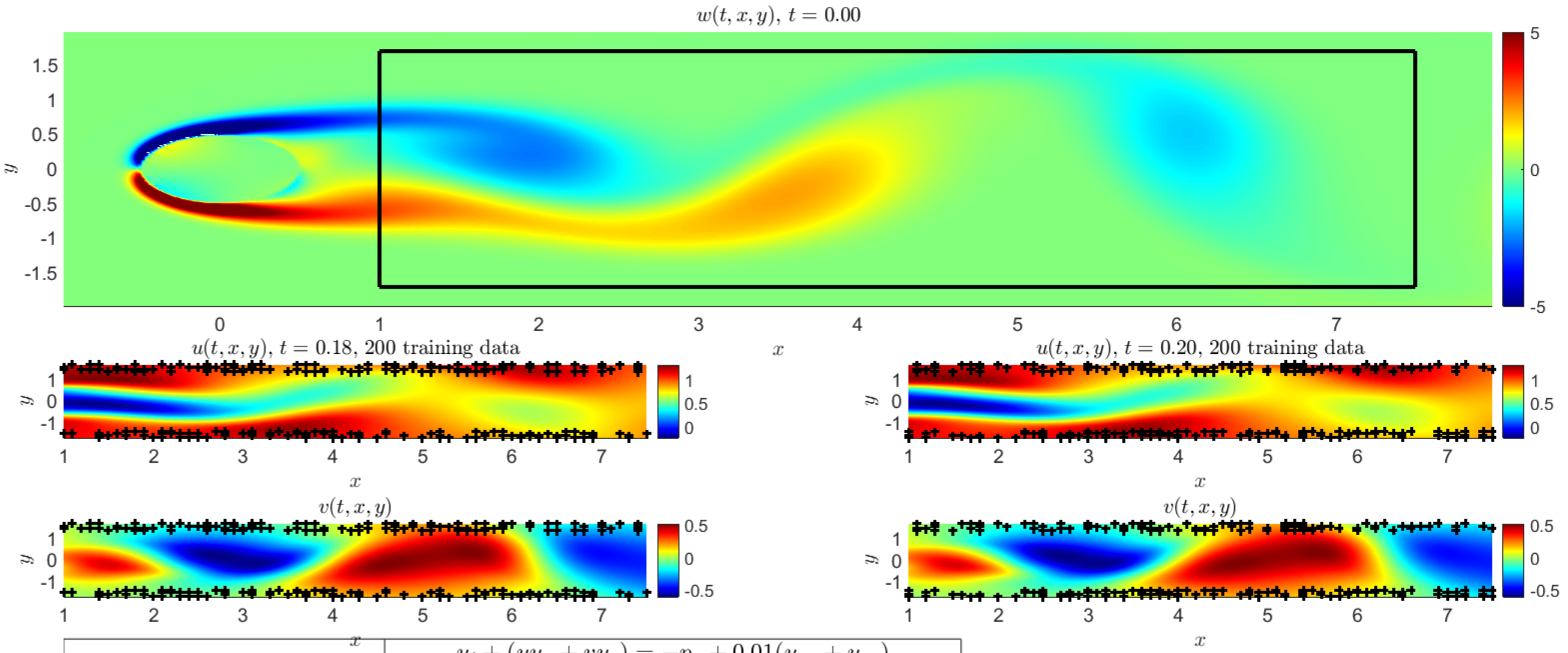
What about the location of the data?

NS Equations
$u_t + \lambda_1 (u \cdot \nabla) u - \lambda_2 \Delta u$
$+ \nabla p = 0$
$\nabla \cdot u = 0$

N = 200, t = 0.18

Training data on the boundary

Fine estimates



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 1.005(uu_x + vu_y) = -p_x + 0.00877(u_{xx} + u_{yy})$ $v_t + 1.005(uv_x + vv_y) = -p_y + 0.00877(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.718(uu_x + vu_y) = -p_x + 0.00472(u_{xx} + u_{yy})$ $v_t + 0.718(uv_x + vv_y) = -p_y + 0.00472(v_{xx} + v_{yy})$

What about the location of the data?

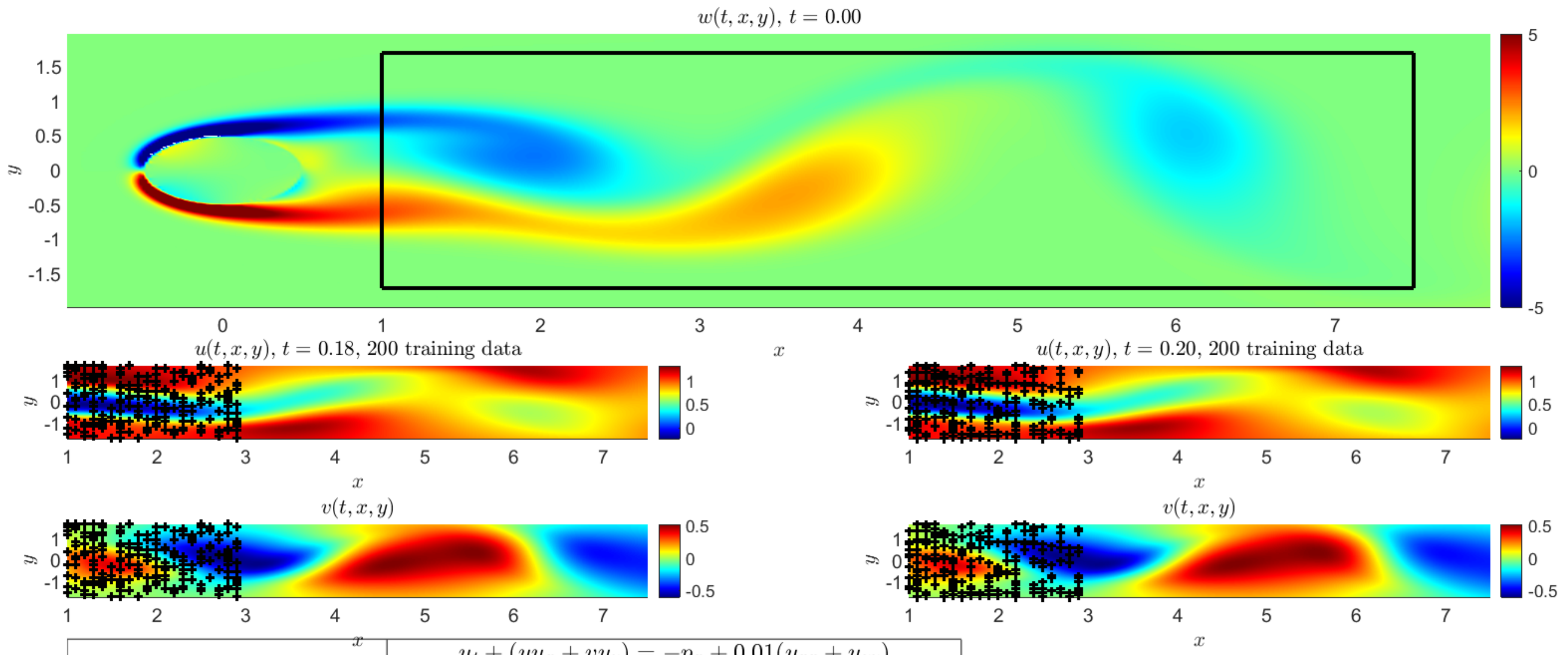
NS Equations

$u_t + \lambda_1 (u \cdot \nabla) u - \lambda_2 \Delta u$
$+ \nabla p = 0$
$\nabla \cdot u = 0$

N = 200, t = 0.18

Training data on the left side of the domain

Good estimates



Correct PDE	$u_t + (uu_x + vu_y) = -p_x + 0.01(u_{xx} + u_{yy})$ $v_t + (uv_x + vv_y) = -p_y + 0.01(v_{xx} + v_{yy})$
Identified PDE (clean data)	$u_t + 1.011(uu_x + vu_y) = -p_x + 0.01005(u_{xx} + u_{yy})$ $v_t + 1.011(uv_x + vv_y) = -p_y + 0.01005(v_{xx} + v_{yy})$
Identified PDE (1% noise)	$u_t + 0.901(uu_x + vu_y) = -p_x + 0.00429(u_{xx} + u_{yy})$ $v_t + 0.901(uv_x + vv_y) = -p_y + 0.00429(v_{xx} + v_{yy})$

Gaussian process *vs* MCMC *vs* Deep Learning

GP

- Very few, noisy observation: only two time snapshots
- Minimization process: trade-off between data-fit and model complexity
- Physics underlies the model (Covariance Kernel)

MCMC

- Iterative process: compute the numerical solution starting from the parameters estimated each time
- Need of efficient sampling strategies

Deep learning approach

- Huge amount of data required
- Physics laws are not considered

Gaussian process

- The multi-output Gaussian process (MOGP) modeling approach allows dealing with multiple correlated outputs

- *Key property of GP*: natural regularization mechanism to infer the unknown model parameters from very few data while effectively preventing overfitting

- *Other applications*: GP are used also for solving PDE’s with no need of discretization in space [3], by proper placement of GP priors

REFERENCES

[1] M.RAISSI, G.KARNIADAKIS, Hidden Physics Models: Machine Learning of Nonlinear Partial Differential Equations, arXiv:1708.00588, 2017

[2] M.RAISSI, Deep Hidden Physics Models: Deep Learning of Nonlinear Partial Differential Equations, arXiv:1801.06637, 2018

[3] M.RAISSI, P. PERDIKARIS, G. KARNIADAKIS, Numerical Gaussian Processes for Time-Dependent and Nonlinear Partial Differential Equations, arXiv:1703.10230, 2017

[4] G.PANG, G.KARNIADAKIS, Physics-informed Learning Machines for PDEs: Gaussian Processes versus Neural Networks, Emerging Frontiers in Nonlinear Science, 2020